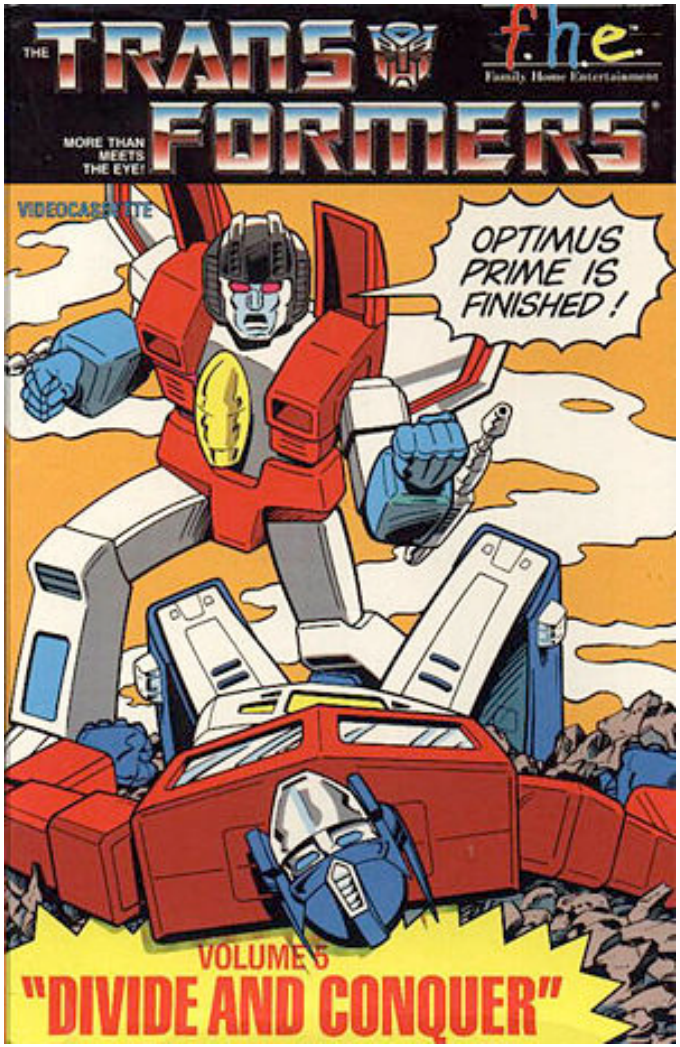
Lecture 2: Divide-and-Conquer!

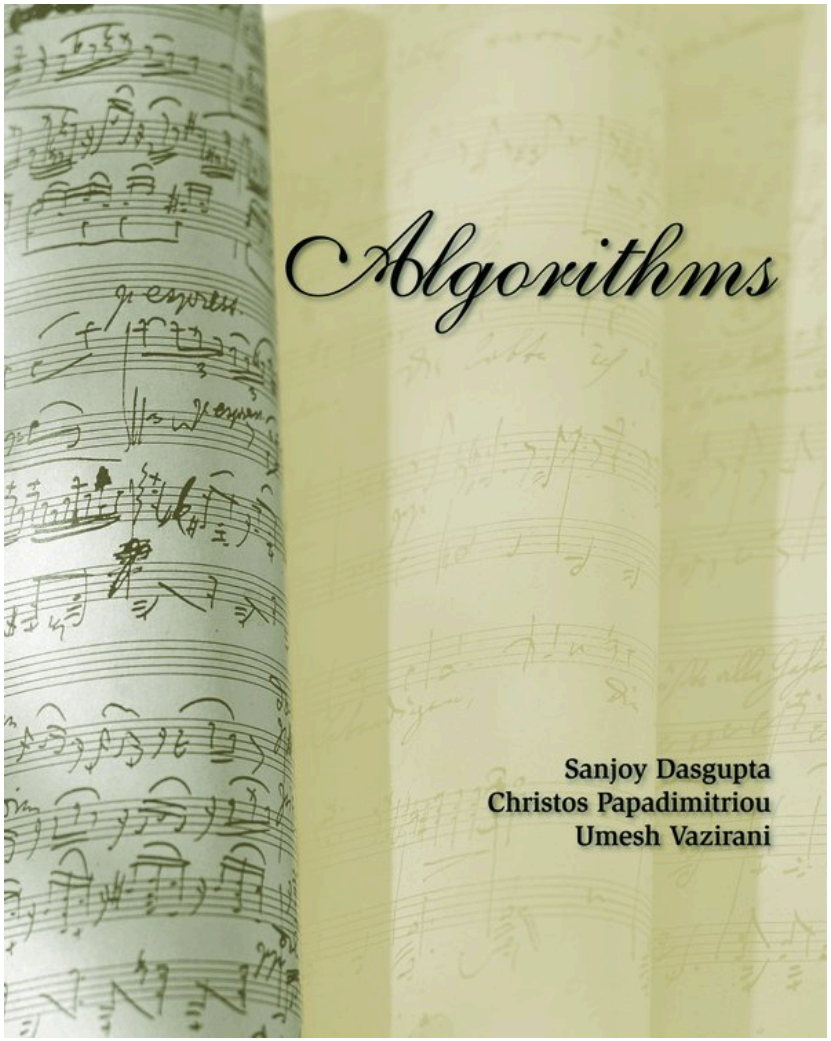


Office hours and grutor hours

- Ran's office hours begin on Friday
  - 4-5:30 PM, Olin 1253
  - Friday office hours begin with **"Algs week in Review"**
- Grutoring begins on Sunday (in this room!)
  - Hours posted on Canvas Syllabus

Everyone can thrive in this course and it's *our* goal to help you achieve *your* goals for this course!

The book


Algorithms
Sanjoy Dasgupta
Christos Papadimitriou
Umesh Vazirani

Last time: The Greedy Paradigm

- It often *doesn't* work!
- But when it does work, the algorithm is usually very simple and very fast

For what denominations does the greedy algorithm give optimal solutions?

Optimal Bounds for the Change-Making Problem

Dexter Kozen[1] and Shmuel Zaks[2]

[1] Computer Science Department, Cornell University
Ithaca, New York 14853, USA
kozen@cs.cornell.edu

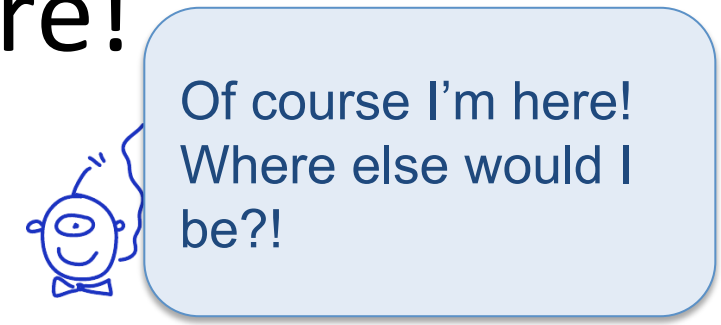
[2] Computer Science Department, Technion
Haifa, Israel
zaks@cs.technion.ac.il

Abstract. The *change-making problem* is the problem of representing a given value with the fewest coins possible. We investigate the problem of determining whether the greedy algorithm produces an optimal representation of all amounts for a given set of coin denominations

Fibonacci denominations work!

Safety and Induction Revisited

You are Here!



- Greedy paradigm
- Divide-and-conquer paradigm
- Dynamic Programming (DP) paradigm
- Amortization paradigms

- Graph algorithms
- "Advanced" topics
  - NP-completeness
  - Online algorithms
  - Parallel algorithms
  - Computational Geometry

HW 1…

1. Properties of logs

a. $\log_b a^n = n \log_b a$.

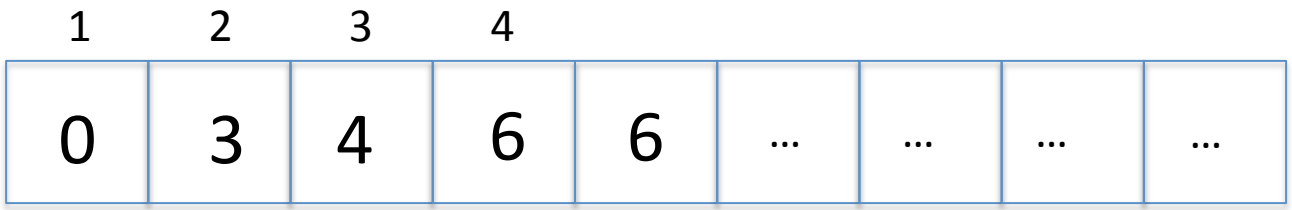
b. $\log_b a = \frac{\log_c a}{\log_c b}$.

c. $a^{\log_b n} = n^{\log_b a}$. (This result is sometimes called the "log-switching theorem" since it says that we can "switch" a and n.)

2. Greed!
3. Greed!!
4. Greed!!!

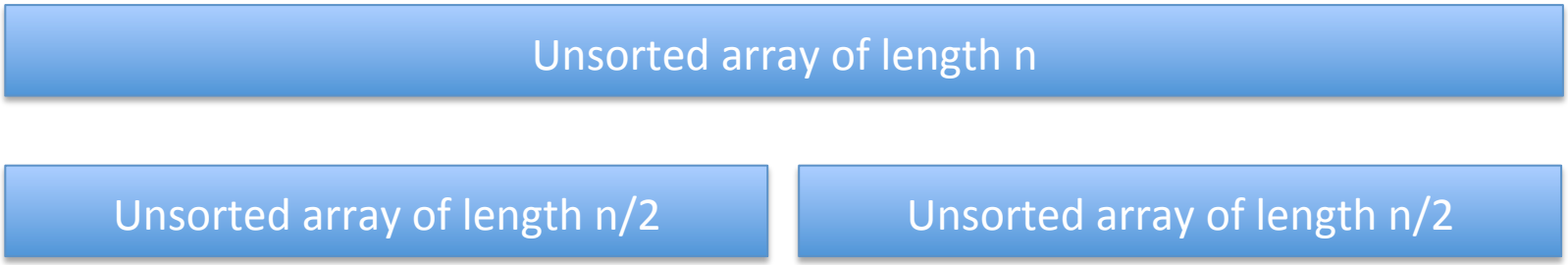
Mergesort

Mergesort



Mergesort

Mergesort

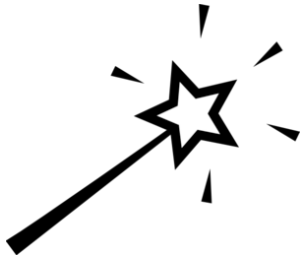

Unsorted array of length n
Unsorted array of length n/2
Unsorted array of length n/2

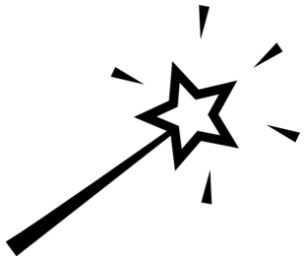
Mergesort

Unsorted array of length n

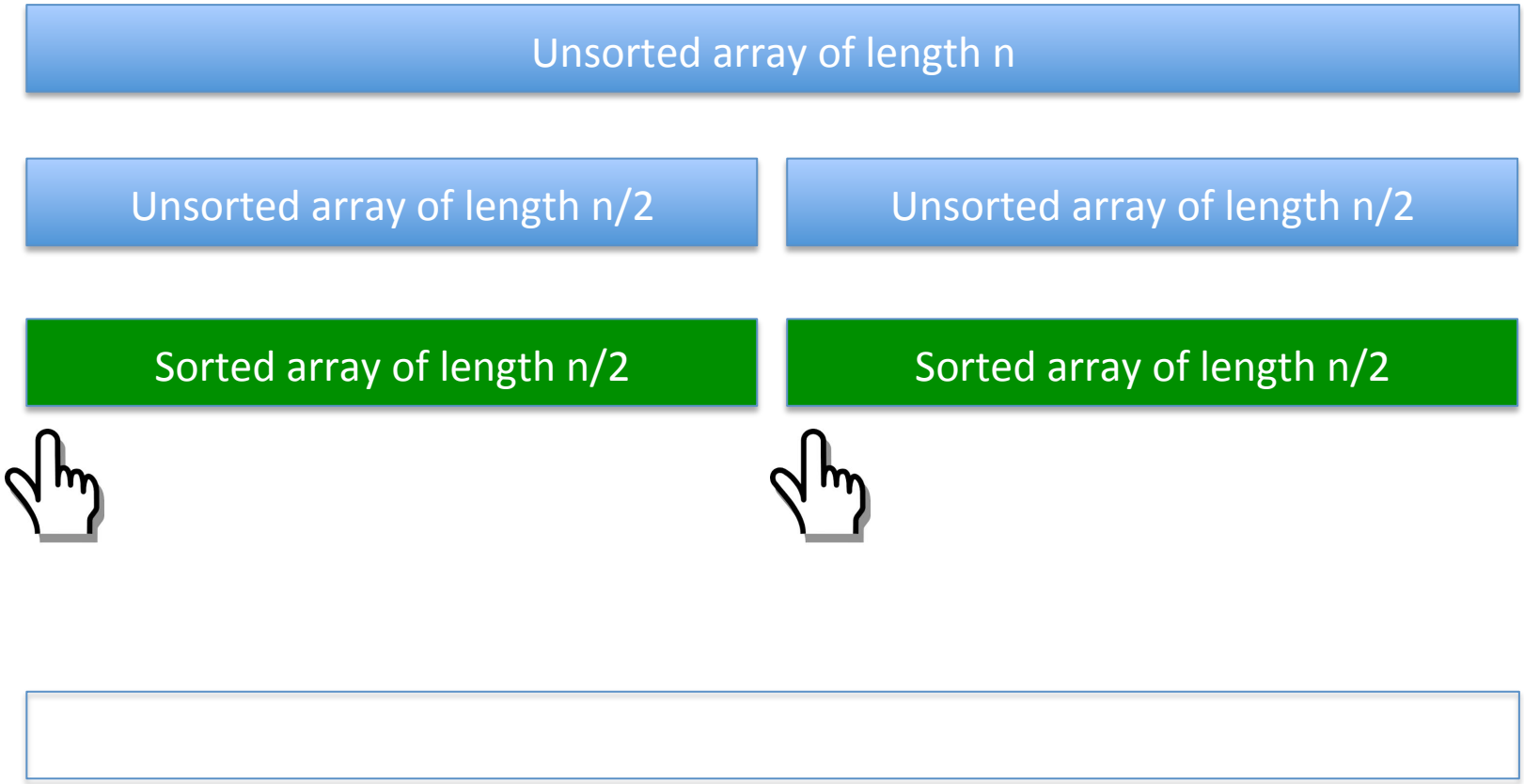
Unsorted array of length n/2

Unsorted array of length n/2

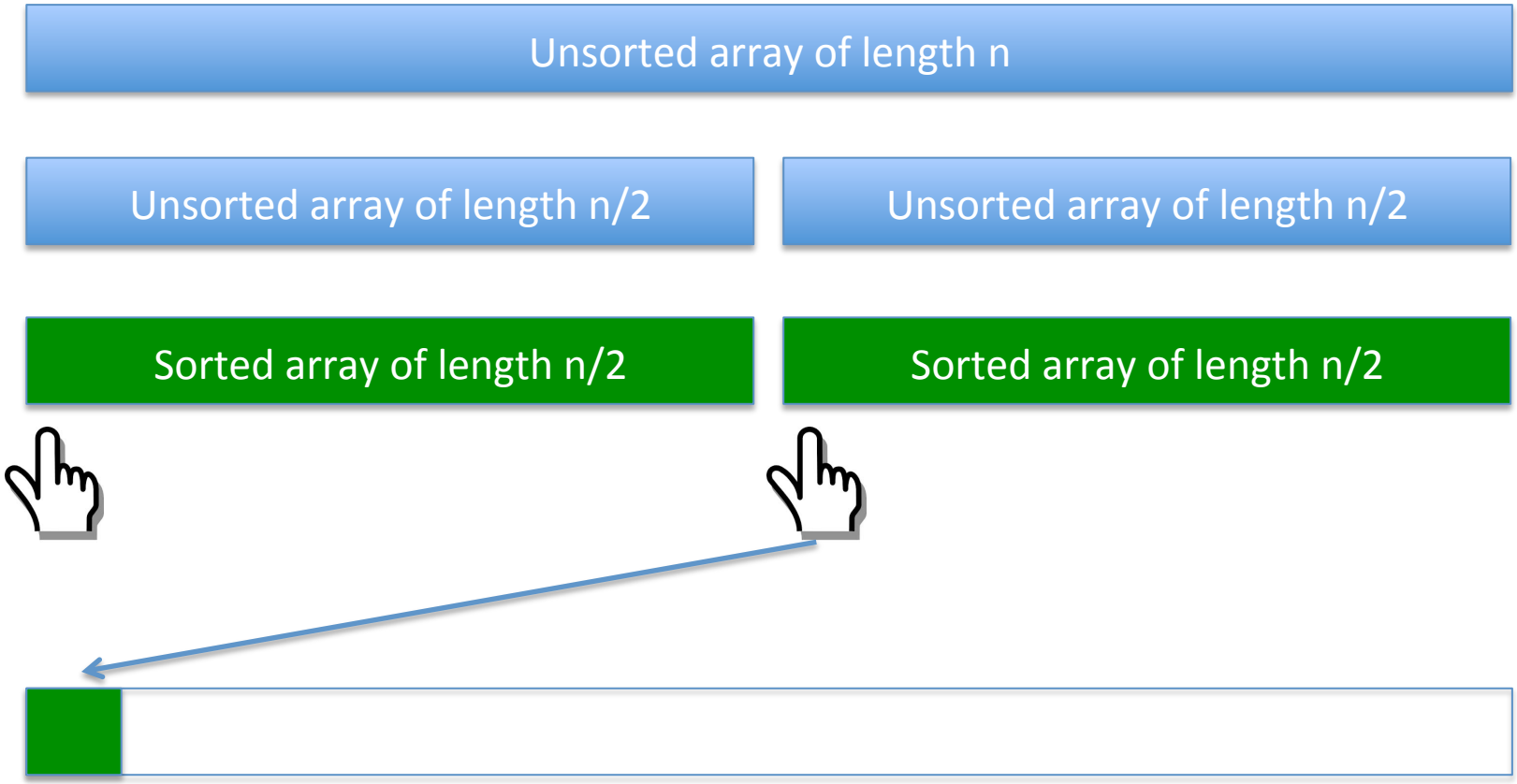




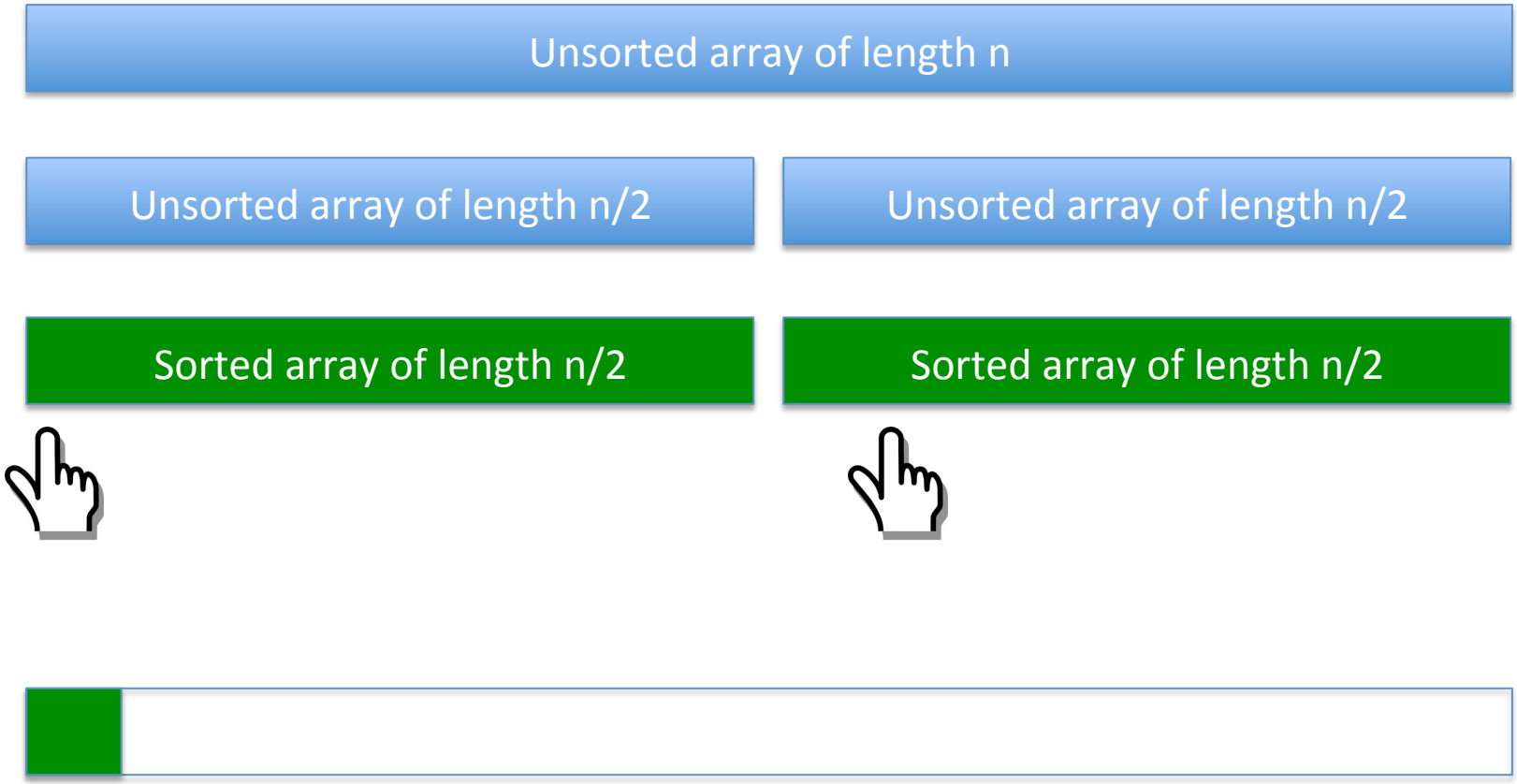
Mergesort


Unsorted array of length n
Unsorted array of length n/2
Unsorted array of length n/2
Sorted array of length n/2
Sorted array of length n/2

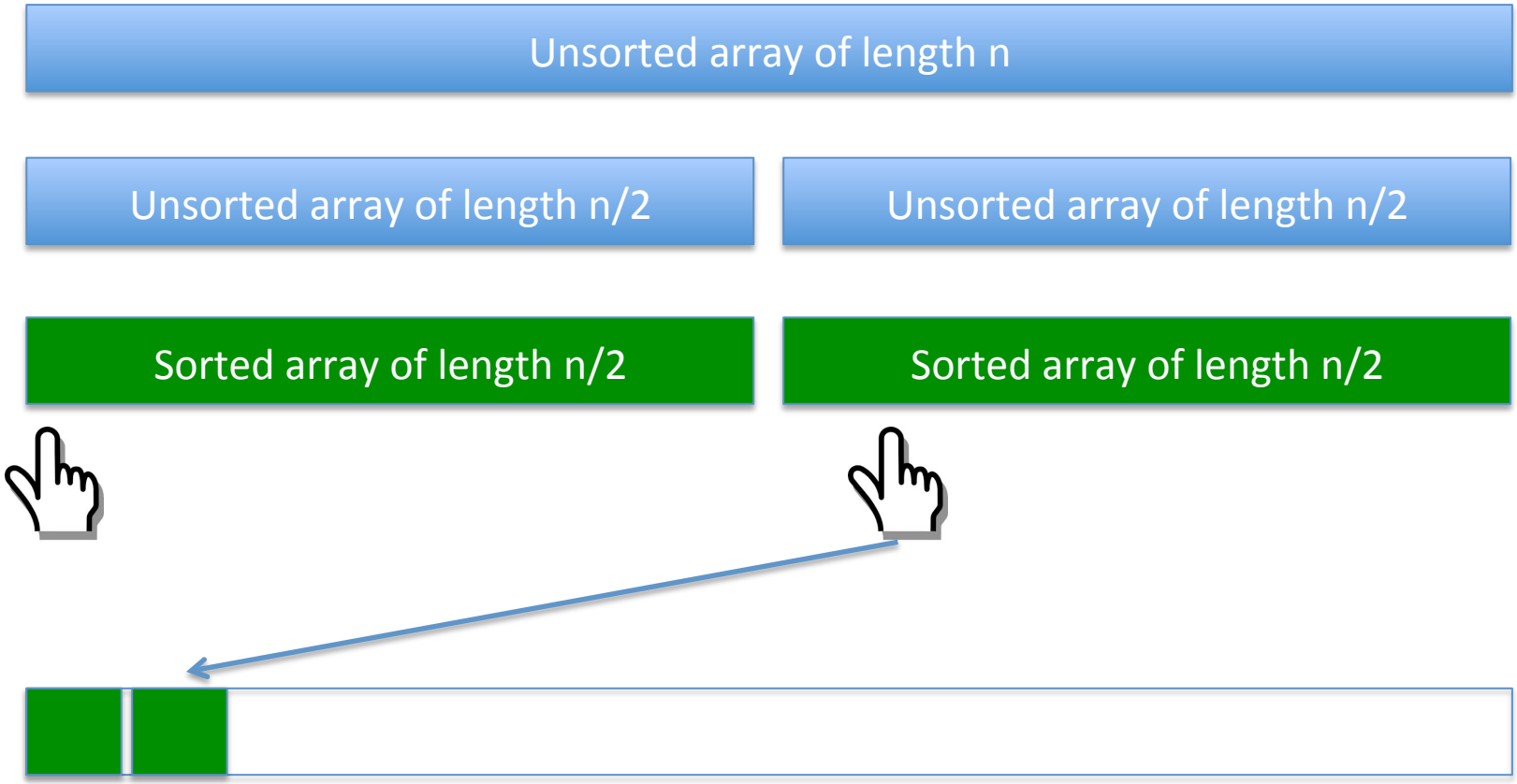
Mergesort


Unsorted array of length n
Unsorted array of length n/2
Unsorted array of length n/2
Sorted array of length n/2
Sorted array of length n/2

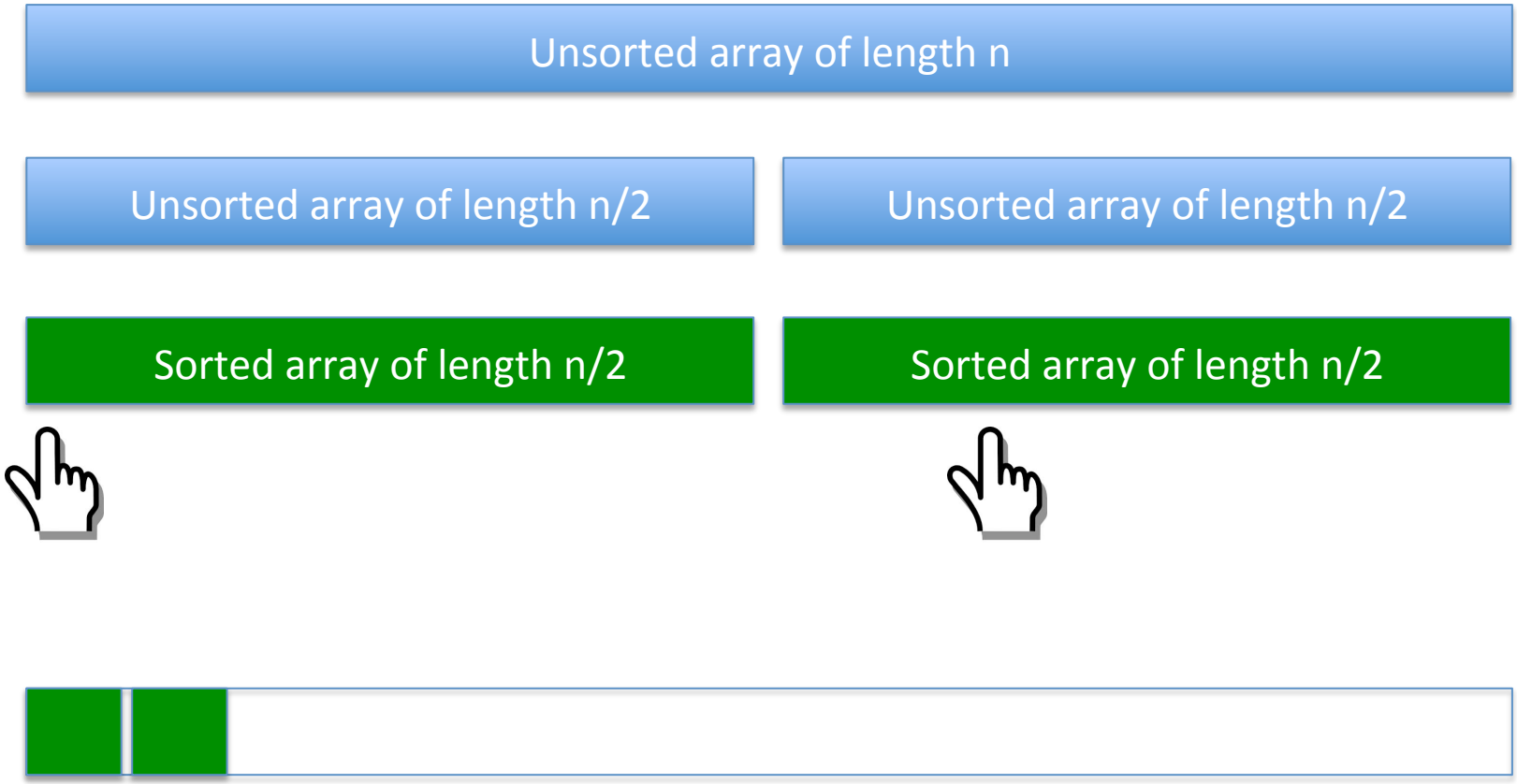
Mergesort


Unsorted array of length n
Unsorted array of length n/2
Unsorted array of length n/2
Sorted array of length n/2
Sorted array of length n/2

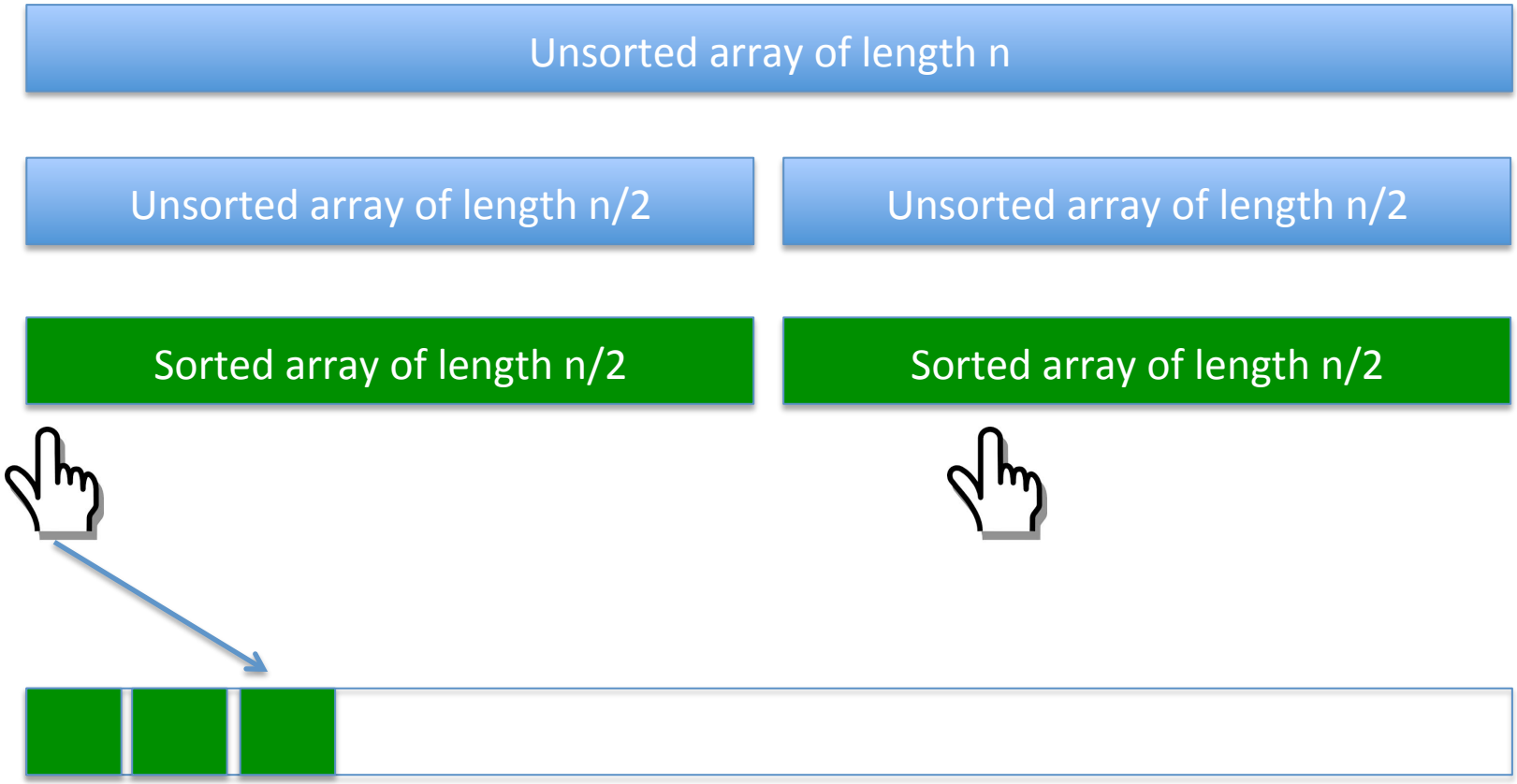
Mergesort


Unsorted array of length n
Unsorted array of length n/2
Unsorted array of length n/2
Sorted array of length n/2
Sorted array of length n/2

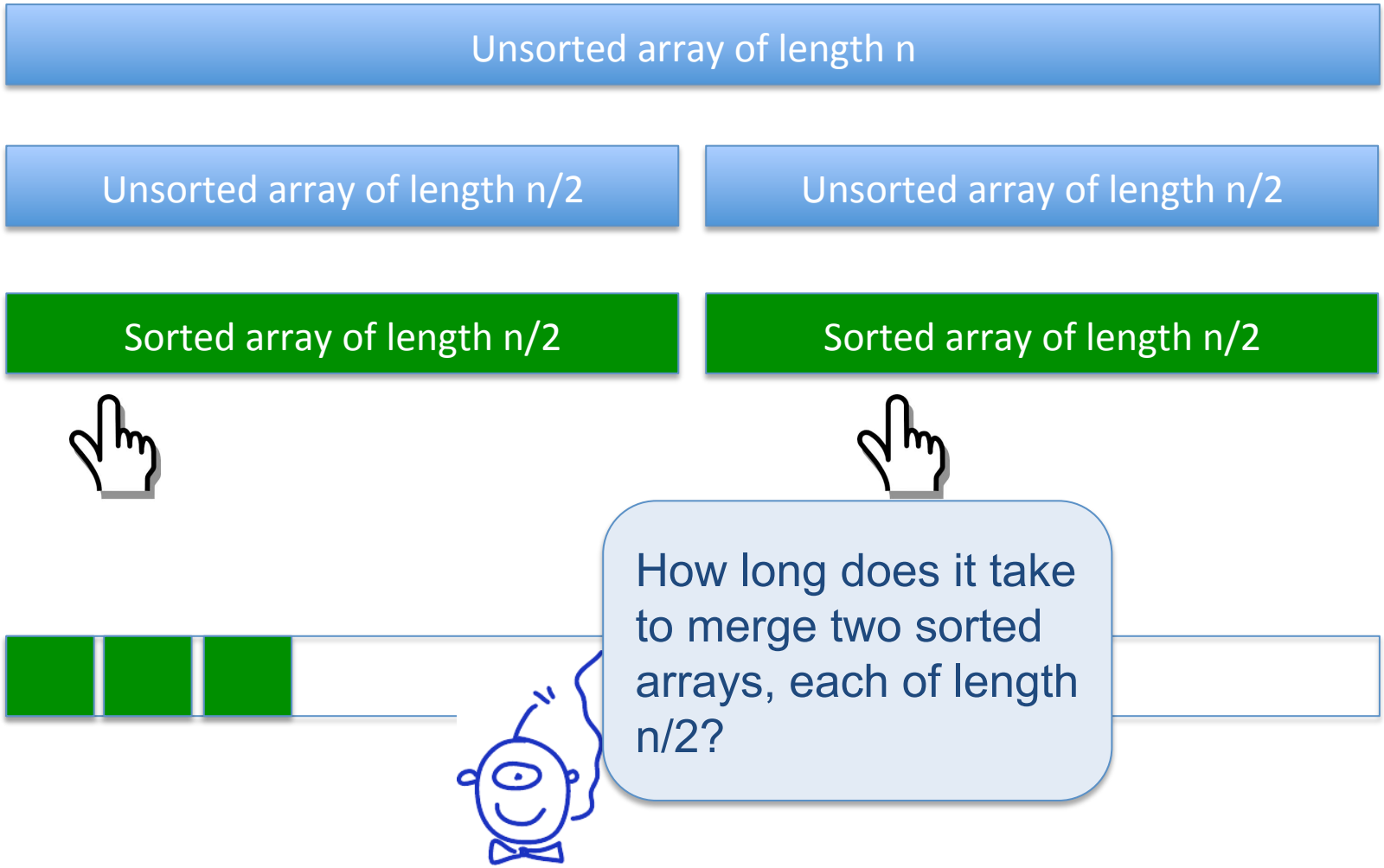
Mergesort


Unsorted array of length n
Unsorted array of length n/2
Unsorted array of length n/2
Sorted array of length n/2
Sorted array of length n/2

Mergesort


Unsorted array of length n
Unsorted array of length n/2
Unsorted array of length n/2
Sorted array of length n/2
Sorted array of length n/2

Mergesort

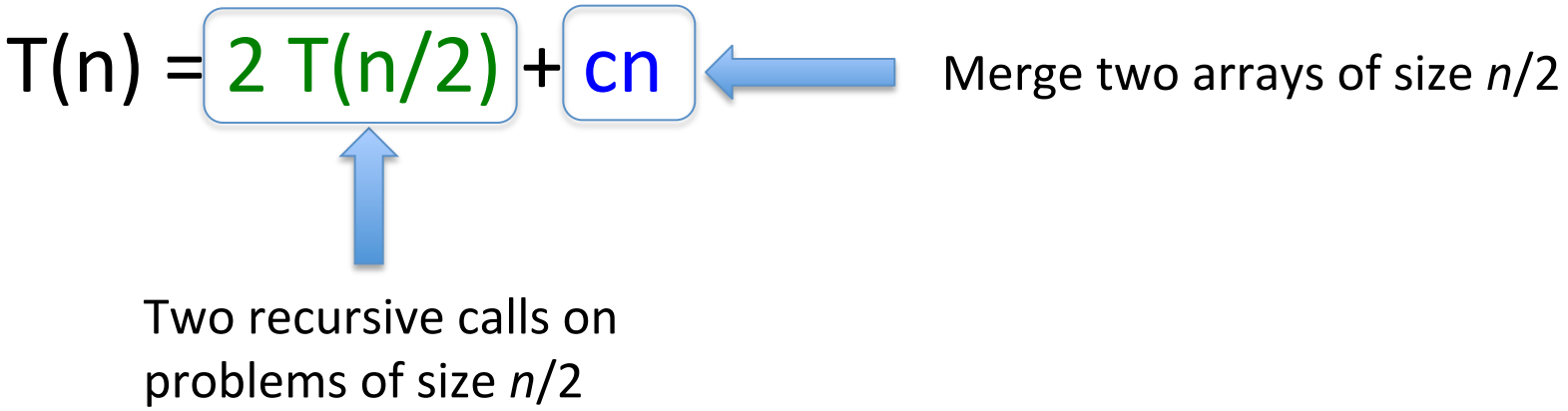

Unsorted array of length n
Unsorted array of length n/2
Unsorted array of length n/2
Sorted array of length n/2
Sorted array of length n/2
How long does it take to merge two sorted arrays, each of length n/2?

Recurrences and Running Times

T(n) = worst case running time of mergesort on input of length *n*

Recurrences and Running Times

T(n) = worst case running time of mergesort on input of length *n*

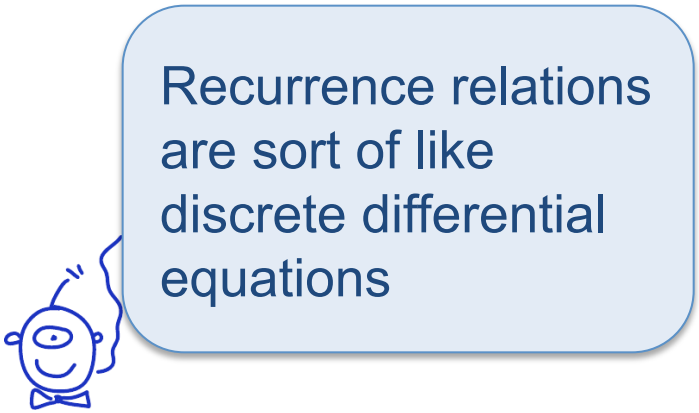


Recurrences and Running Times

T(n) = worst case running time of mergesort on input of length *n*

T(n) = 2 T(n/2) + cn

T(1) = d



Recurrences and Running Times

T(n) = worst case running time of mergesort on input of length *n*

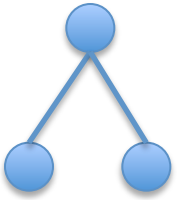
T(n) = 2 T(n/2) + cn Assume $n = 2^k$

T(1) = d

$T(n) = 2\,T(n/2) + cn$ Assume $n = 2^k$

$T(1) = d$

"Recursion Tree" Analysis

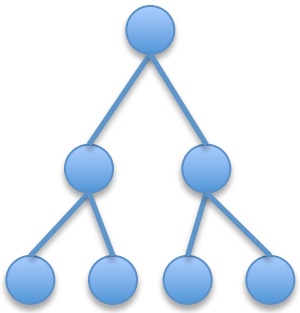


# Nodes	Problem size	Work/node	Total work
1	2^k	$c\,2^k$	$c\,2^k$
2	2^{k-1}	$c\,2^{k-1}$	$2\,c\,2^{k-1}$

$T(n) = 2\,T(n/2) + cn$ Assume $n = 2^k$

$T(1) = d$

"Recursion Tree" Analysis

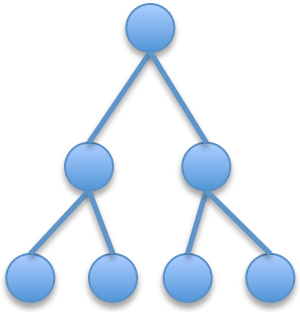


# Nodes	Problem size	Work/node	Total work
1	2^k	$c\,2^k$	$c\,2^k$
2	2^{k-1}	$c\,2^{k-1}$	$2\,c\,2^{k-1}$
2^2	2^{k-2}	$c\,2^{k-2}$	$2^2\,c\,2^{k-2}$

$T(n) = 2\,T(n/2) + cn$ Assume $n = 2^k$

$T(1) = d$

"Recursion Tree" Analysis



# Nodes	Problem size	Work/node	Total work
1	2^k	$c\,2^k$	$c\,2^k$
2	2^{k-1}	$c\,2^{k-1}$	$2\,c\,2^{k-1}$
2^2	2^{k-2}	$c\,2^{k-2}$	$2^2\,c\,2^{k-2}$
⋮	⋮		
2^k	2^{k-k}	d	$2^k\,d\,2^{k-k}$

$T(n) = 2\,T(n/2) + cn$ Assume $n = 2^k$

$T(1) = d$

# Nodes	Problem size	Work/node	Total work
2^0	2^k	$c\,2^k$	$c\,2^k$
2^1	2^{k-1}	$c\,2^{k-1}$	$2\,c\,2^{k-1}$
2^2	2^{k-2}	$c\,2^{k-2}$	$2^2\,c\,2^{k-2}$
⋮	⋮		
2^k	2^{k-k}	d	$2^k\,d\,2^{k-k}$

$$T(n) = c\,(2^k + 2^k + \ldots + 2^k) + d2^k$$

How many terms here?

$T(n) = 2\,T(n/2) + cn$ Assume $n = 2^k$

$T(1) = d$

# Nodes	Problem size	Work/node	Total work
1	2^k	$c\,2^k$	$c\,2^k$
2	2^{k-1}	$c\,2^{k-1}$	$2\,c\,2^{k-1}$
2^2	2^{k-2}	$c\,2^{k-2}$	$2^2\,c\,2^{k-2}$
⋮	⋮		
2^k	2^{k-k}	d	$2^k\,d\,2^{k-k}$

$$T(n) = c\,(2^k + 2^k + \ldots + 2^k) + d2^k \leq c(n \log n) + dn$$

How many terms here?

Why don't you indicate that this is log base 2?

The "Mergesort Template"

$T(n) = 2T(n/2) + O(n)$

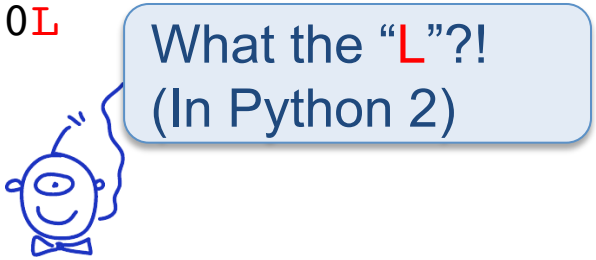
- Divide the problem into two halves
- Recursively solve each half
- Spend O(n) time completing the solution

Examples:

- Finding the convex hull of n points in the plane
- Fast Fourier Transform
- Many others…

Multiplying Big Numbers

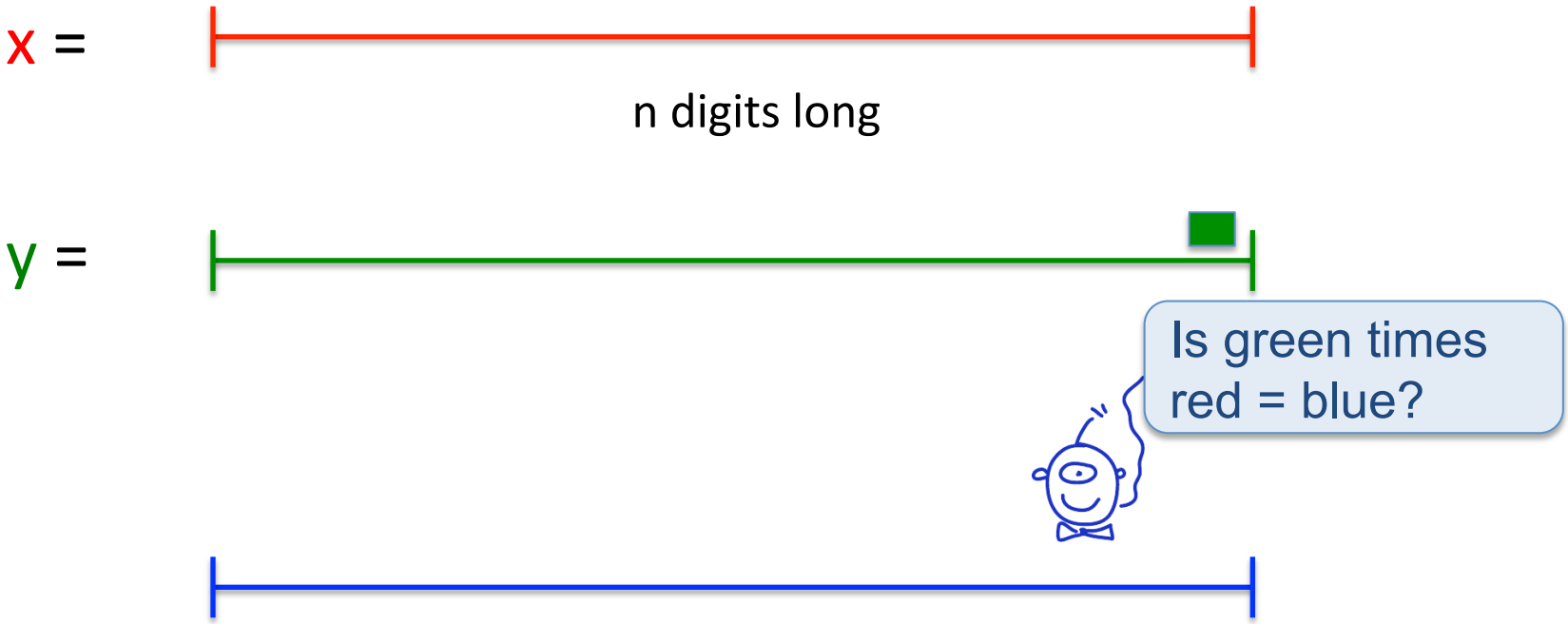
```
% python
>>> x = 12345678910111213141516171819020212223240
>>> y = 60466046604660466046604660466046604660466046
>>> x*y
74649439634495844238030577876683699387021637966327579518486487971315241921090 40L
```

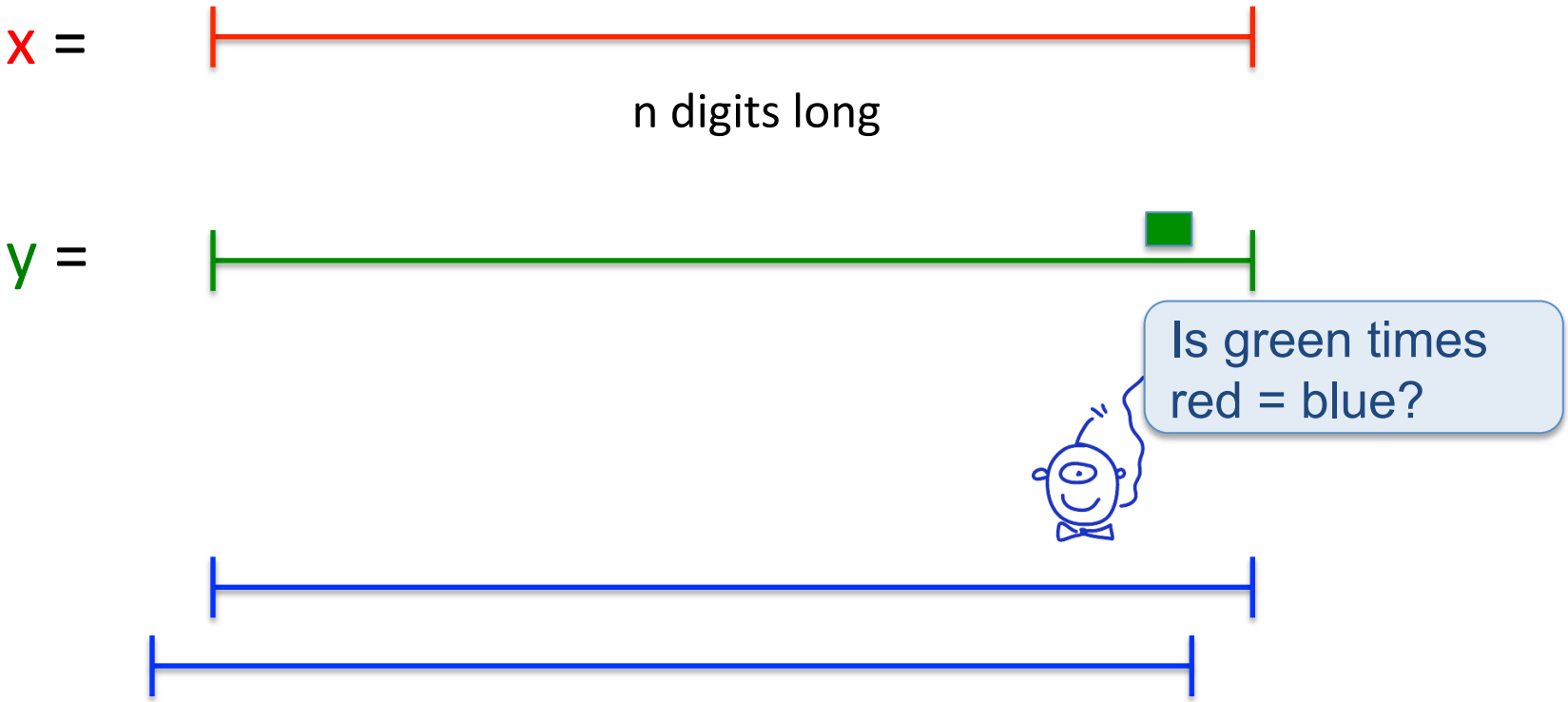


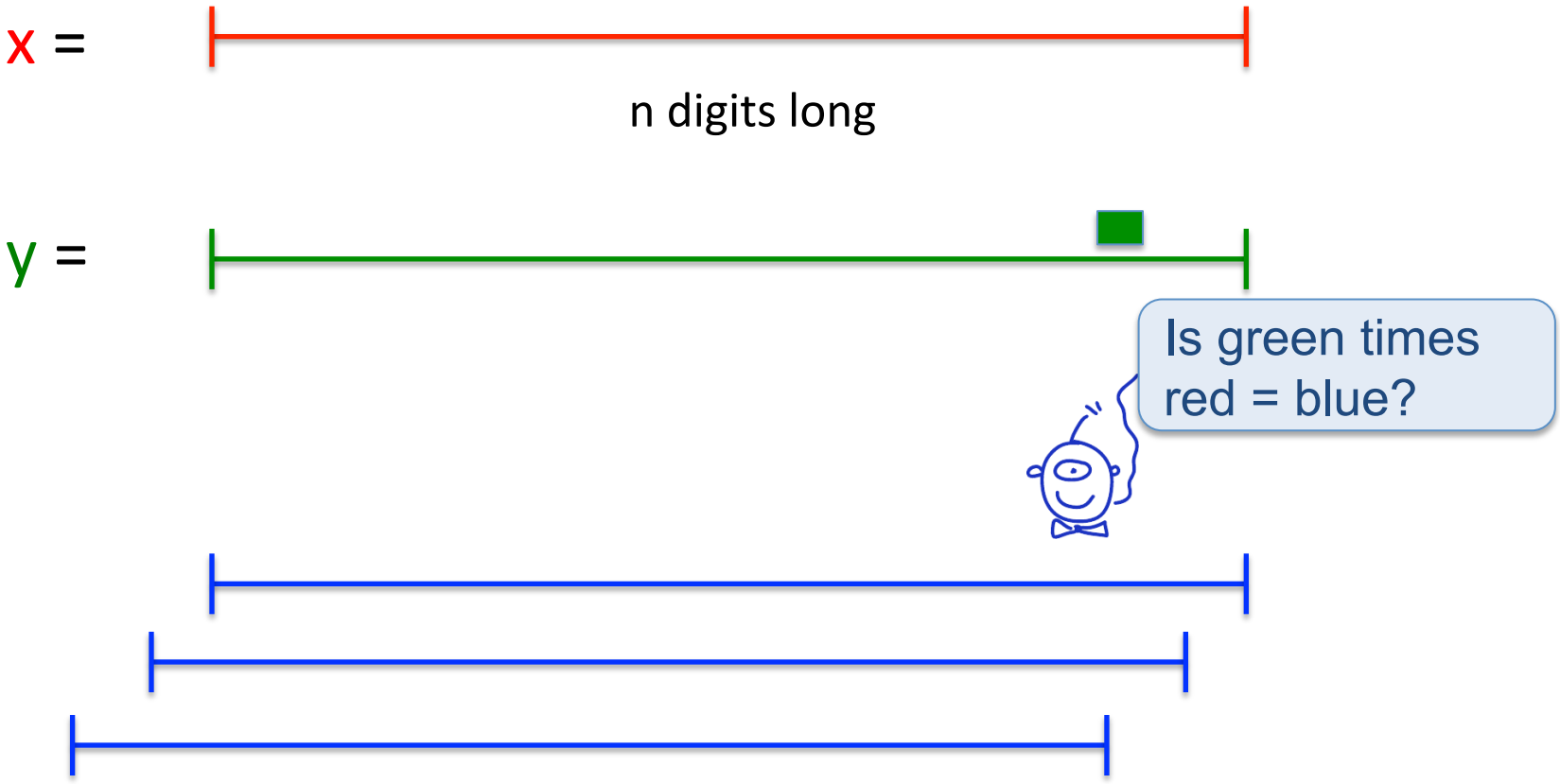
x =

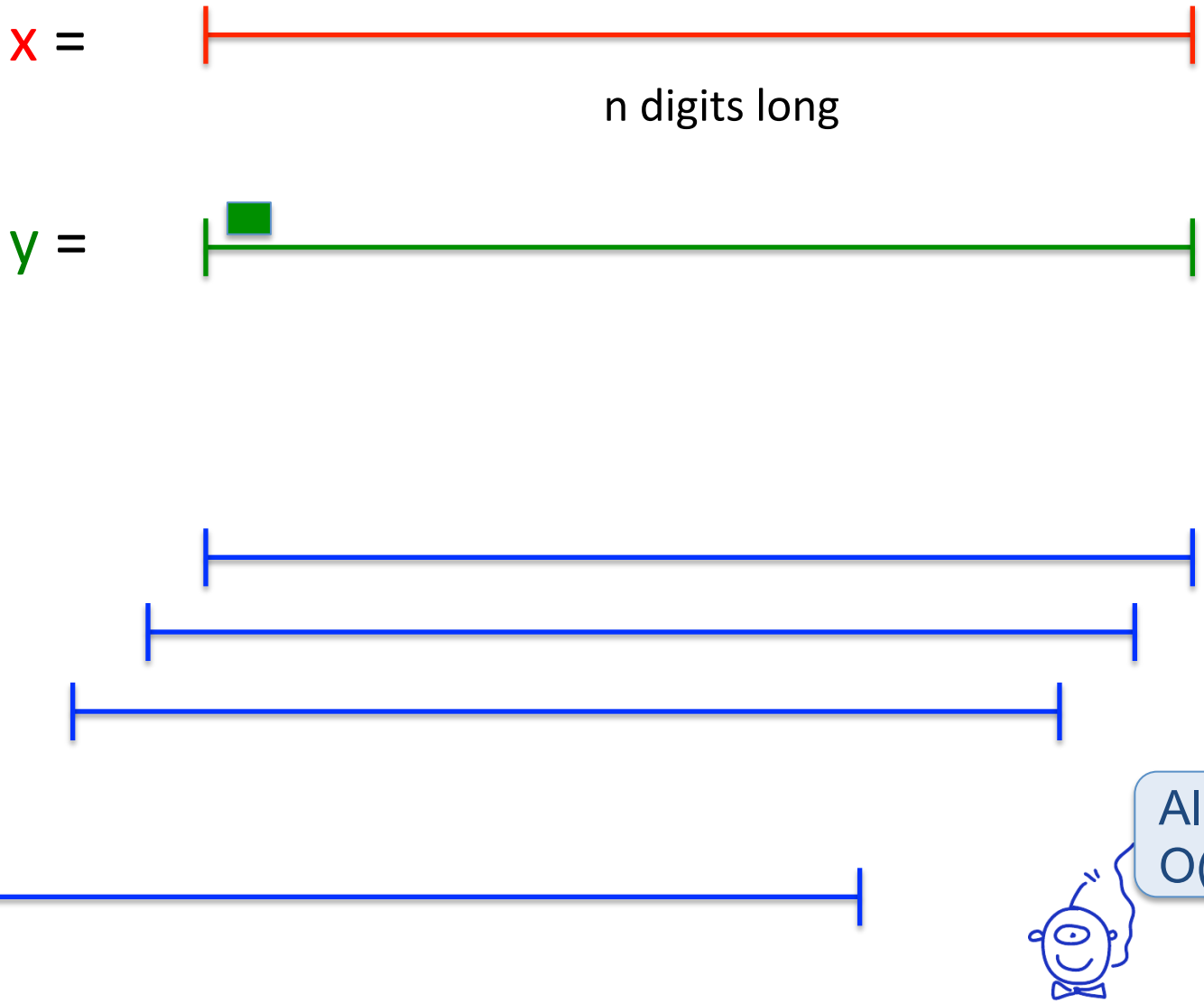
n digits long

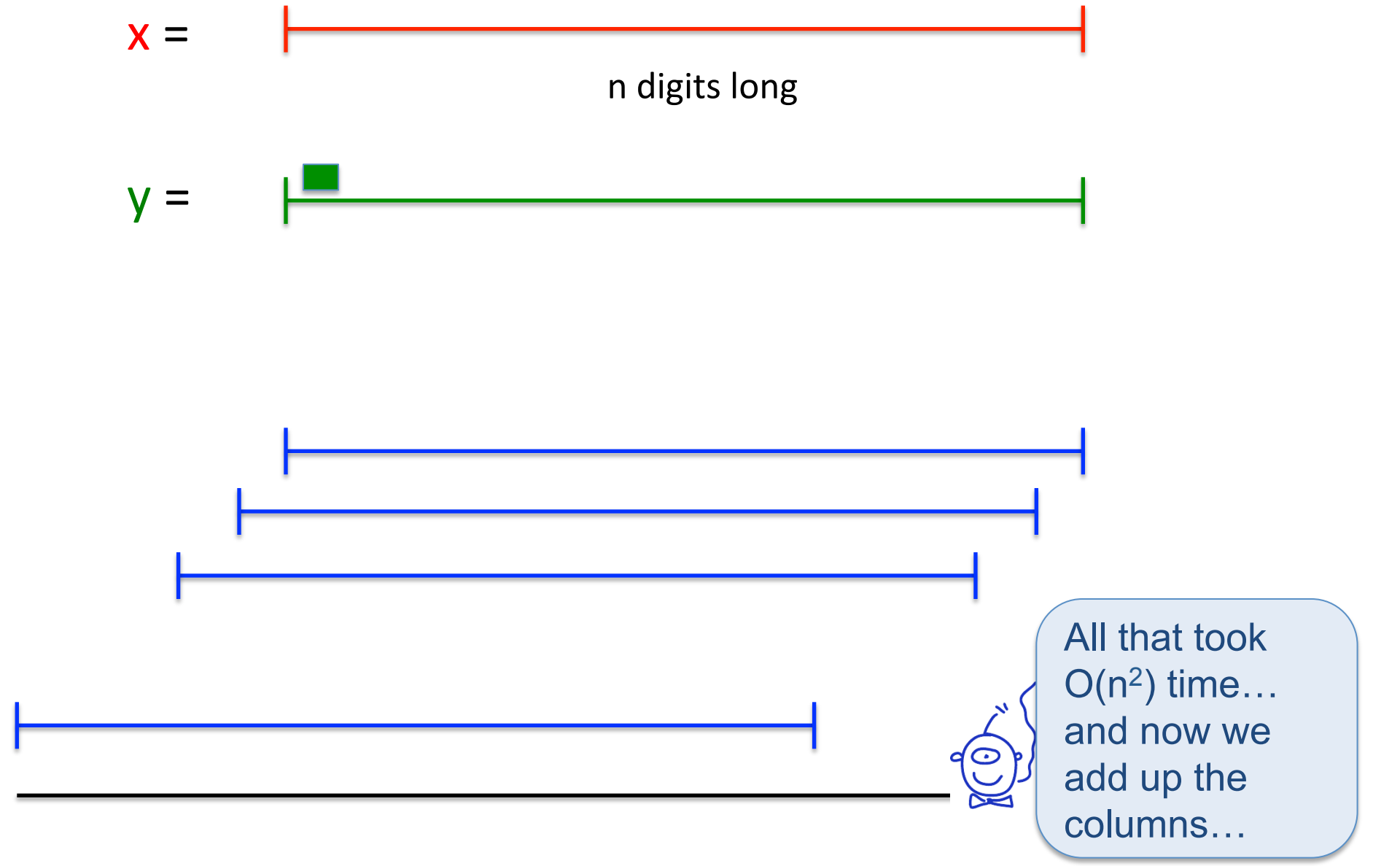
y =

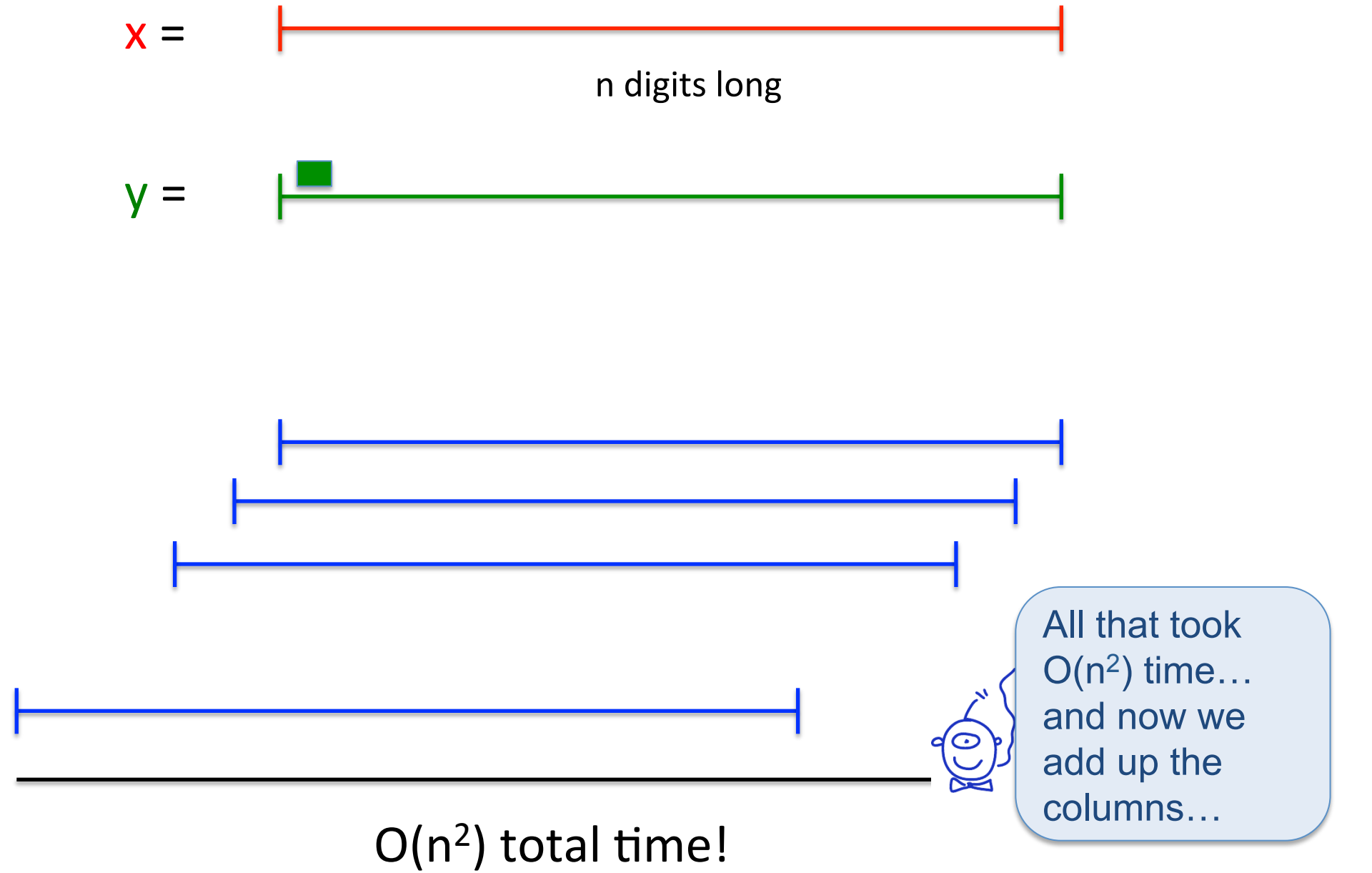

x =
n digits long
y =
Is green times red = blue?


x =
n digits long
y =
Is green times red = blue?


x =
n digits long
y =
Is green times red = blue?


x =
n digits long
y =
All that took
O(n2) time


x =
n digits long
y =
All that took O(n²) time… and now we add up the columns…


x =
n digits long
y =
All that took O(n²) time… and now we add up the columns…
O(n²) total time!

Karatsuba!

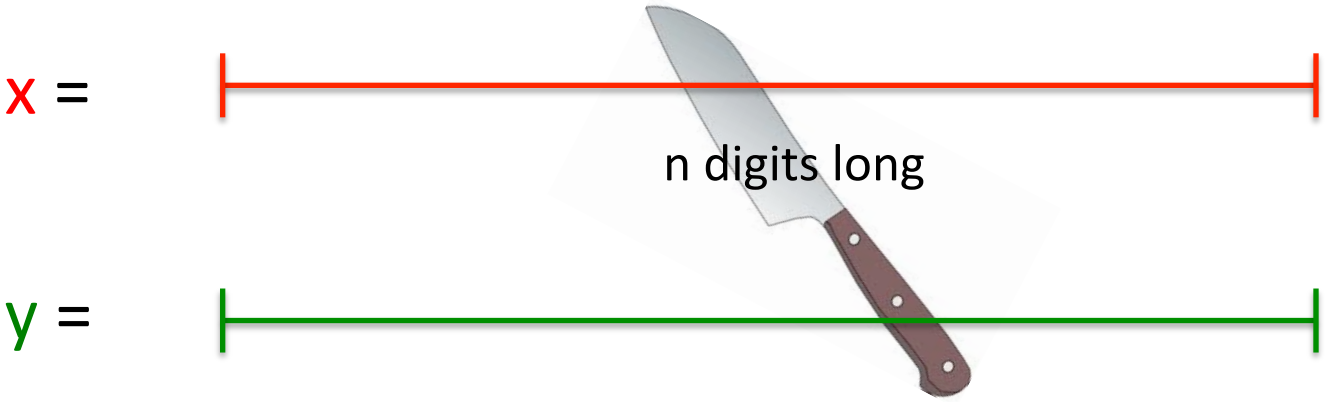
Anatoly Karatsuba

x =

n digits long

y =

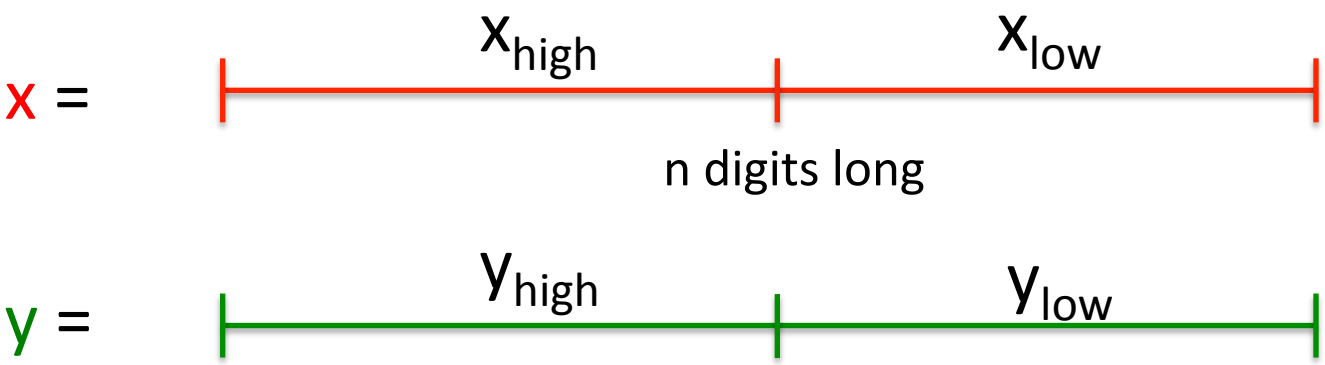
Karatsuba!



Anatoly Karatsuba

Karatsuba!

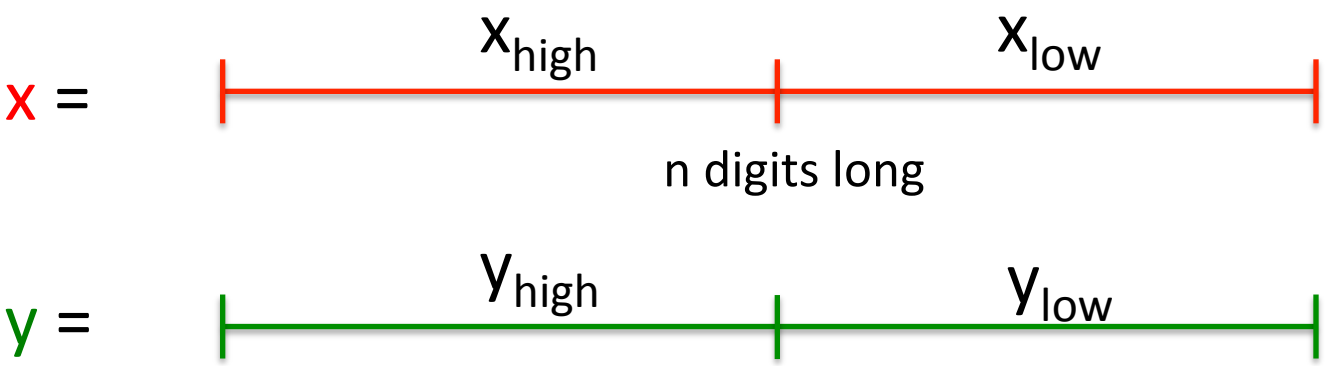
Anatoly Karatsuba



$$xy = (x_{high}\, 10^{n/2} + x_{low})(y_{high}\, 10^{n/2} + y_{low}) =$$
$$x_{high}\, y_{high}\, 10^{n} + (x_{high} y_{low} + x_{low} y_{high})\, 10^{n/2} + x_{low} y_{low}$$

Karatsuba!

Anatoly Karatsuba



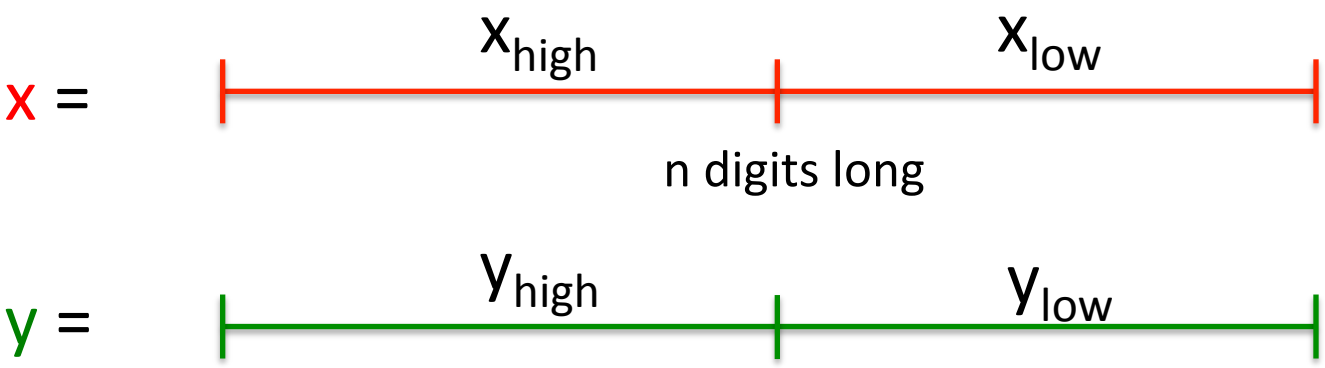
$$xy = (x_{high}\ 10^{n/2} + x_{low})(y_{high}\ 10^{n/2} + y_{low}) =$$

$$x_{high}\ y_{high}\ 10^{n} + (x_{high}y_{low} + x_{low}y_{high})\ 10^{n/2} + x_{low}y_{low}$$

Recursive calls on problems of length n/2

Karatsuba!

Anatoly Karatsuba



$$xy = (x_{high}\,10^{n/2} + x_{low})(y_{high}\,10^{n/2} + y_{low}) =$$

$$x_{high}\,y_{high}\,10^{n} + (x_{high}y_{low} + x_{low}y_{high})\,10^{n/2} + x_{low}y_{low}$$

Recursive calls on problems of length n/2

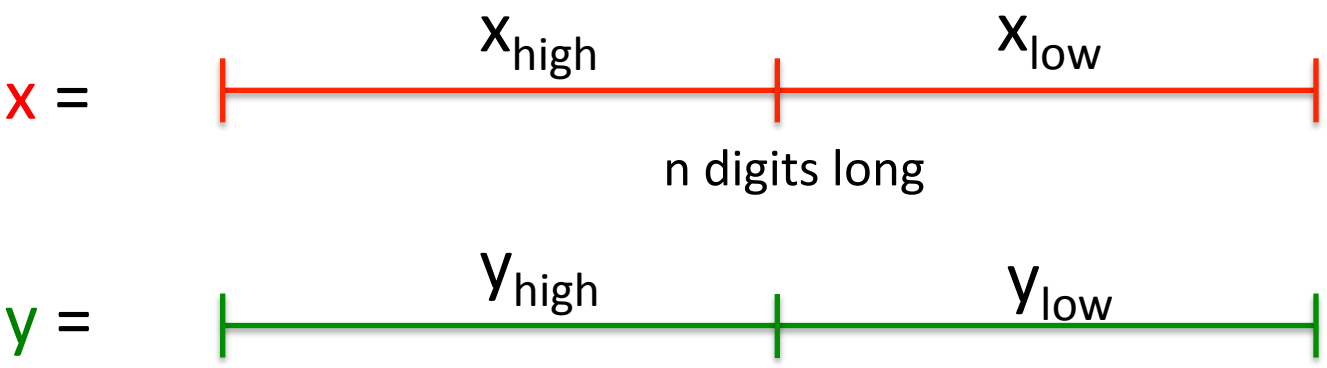
What about these multiplications?

And what about the additions?

This Space for Rent

Karatsuba, Part Deux

Anatoly Karatsuba



$$xy = (x_{high}\, 10^{n/2} + x_{low})(y_{high}\, 10^{n/2} + y_{low}) =$$
$$x_{high}\, y_{high}\, 10^{n} + (x_{high}y_{low} + x_{low}y_{high})\, 10^{n/2} + x_{low}y_{low}$$

Alternatively…

- Let $P_1 = x_{high}\, y_{high}$
- Let $P_2 = x_{low}\, y_{low}$
- Let $P_3 = (x_{high} + x_{low})(y_{high} + y_{low})$
- Now, $x_{high}y_{low} + x_{low}y_{high} = P_3 - P_1 - P_2$

Worksheet: Try "fixing" the recurrence relation and then solve it!

More space for rent!

Big-Oh

$$3n^2 + 5 \in O(n^2)$$

$$n \log n \in O(n^2)$$

$$2n + 1 \in O(n^2)$$

$$42 \in O(n^2)$$

Which of these statements is true? (And what's with the "element of" notation?!)

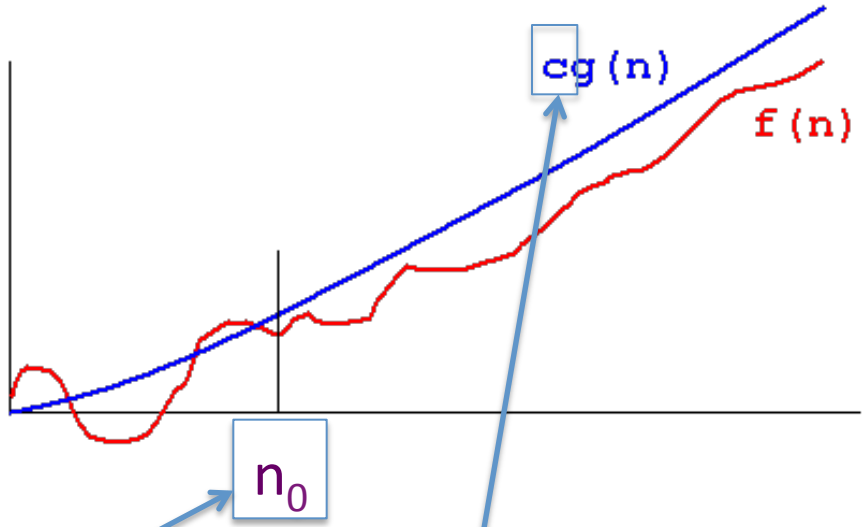
f(n) g(n)

$$3n^2 + 5 \in O(n^2)$$

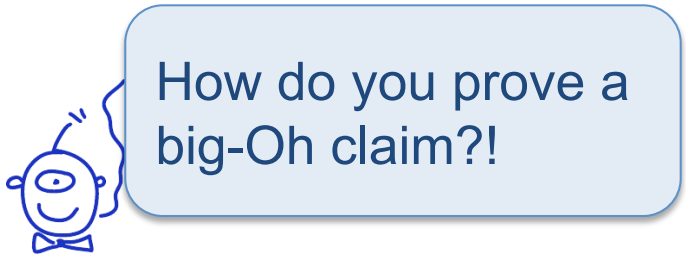
$$n \log n \in O(n^2)$$

$$2n + 1 \in O(n^2)$$

$$42 \in O(n^2)$$



$$f(n) \in O(g(n)) \text{ iff } \exists c > 0, n_0 \geq 0 \text{ s.t. } f(n) \leq cg(n) \ \forall n \geq n_0$$



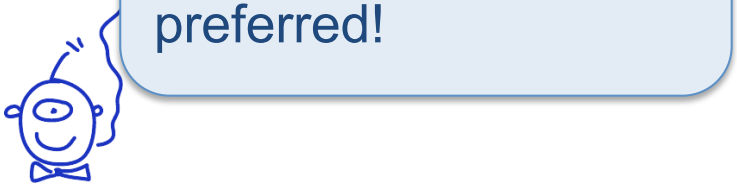
f(n) g(n)

$$3n^2 + 5 \in O(n^2)$$

$$n \log n \in O(n^2)$$

$$2n + 1 \in O(n^2)$$

$$42 \in O(n^2)$$



$$f(n) \in O(g(n)) \text{ iff } \exists c > 0, n_0 \geq 0 \text{ s.t. } f(n) \leq cg(n) \ \forall n \geq n_0$$

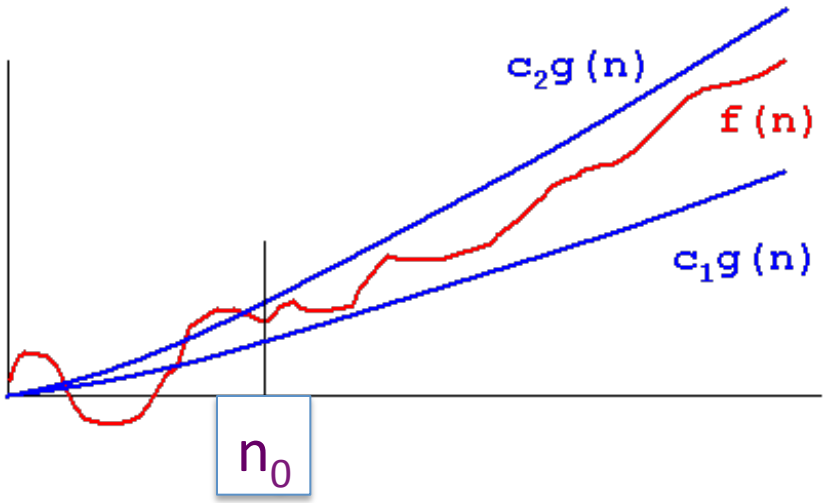
f(n) g(n)

$$3n^2 + 5 \in O(n^2)$$

$$n \log n \in O(n^2)$$

$$2n + 1 \in O(n^2)$$

$$42 \in O(n^2)$$



$$f(n) \in \Theta(g(n)) \text{ iff } \exists c_1, c_2 > 0, n_0 \geq 0 \text{ s.t.}$$
$$c_1 g(n) \leq f(n) \leq c_2 g(n) \ \forall n \geq n_0$$

My call with Prof. I. Lai this morning…

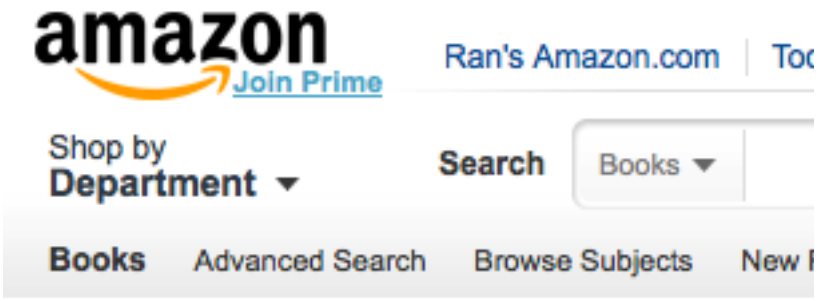
I claim that…

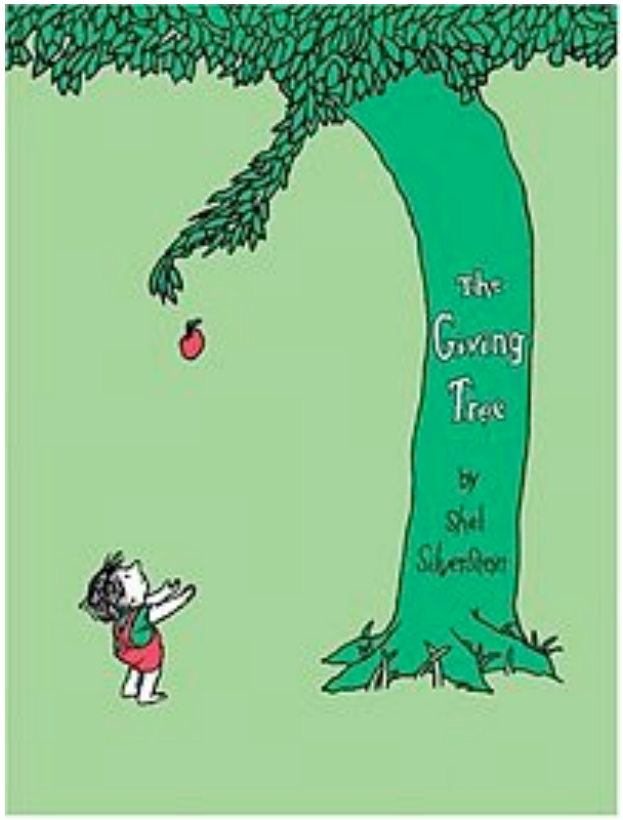
$$42 \in O(1)$$

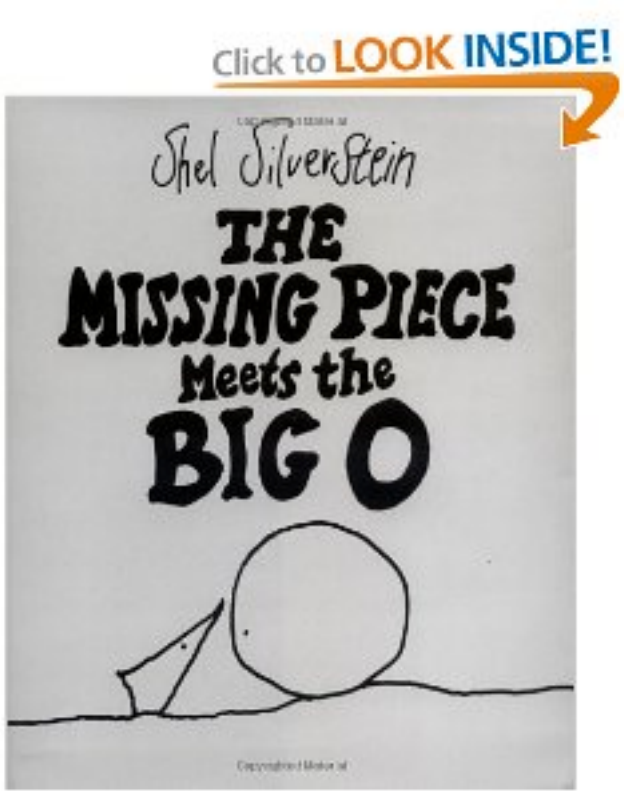
$$1.0001^n \in O(n^{42})$$

$$f(x) \in O(g(x)) \text{ if } \exists c > 0, n_0 \geq 0 \text{ st}$$
$$f(n) \leq cg(n) \forall n \geq n_0$$

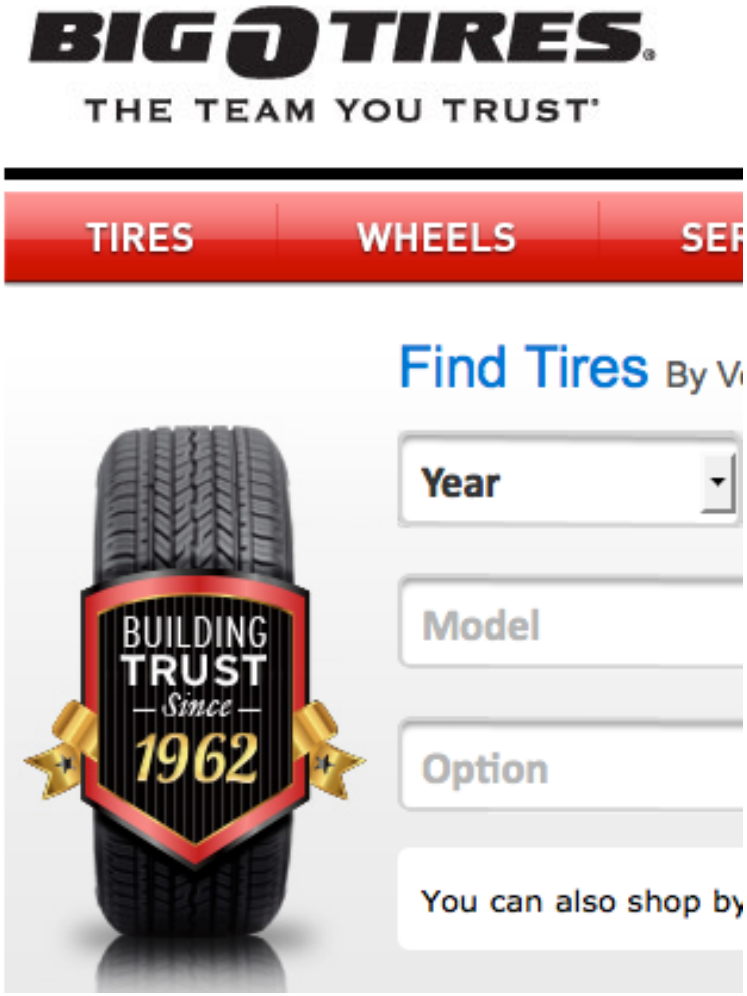
Algorithms books for kids…





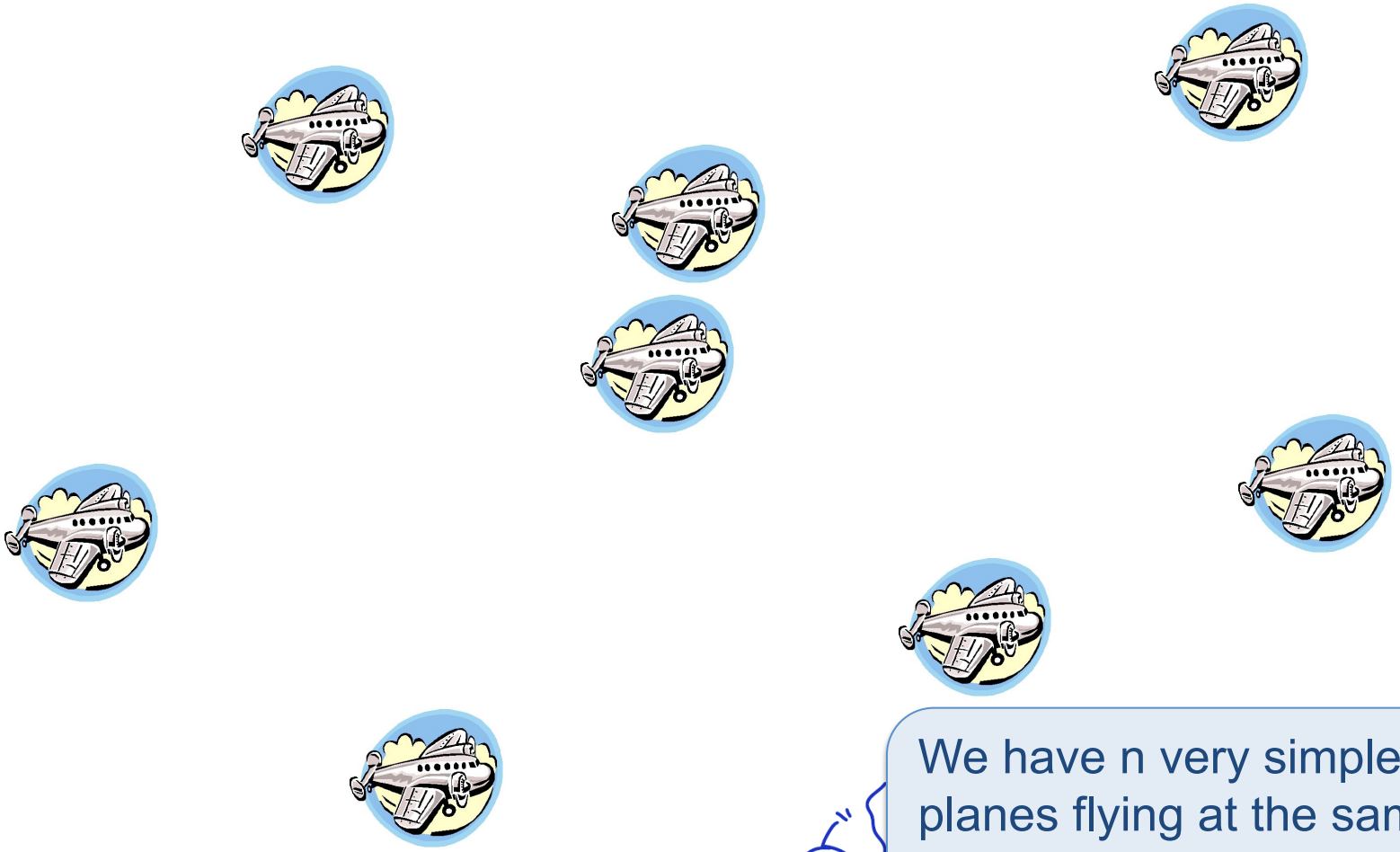


No more than O(1) flat tires per year!



Is this TV series O(good)?

Closest Pair of Points



The Cast of Characters…

Matrix Multiplication!

Volker Strassen

Don Coppersmith

Shmuel Winograd

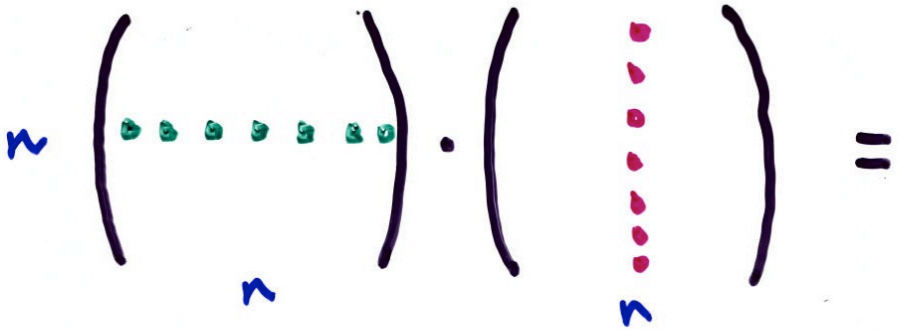
Virginia V. Williams

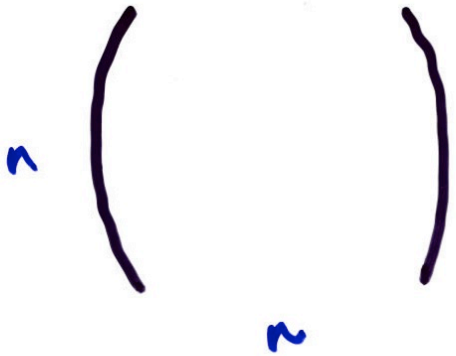
Matrix Multiplication

(starring Arnold Schwarzenegger as The Determinator)

Rated O(PG)

(Let's consider matrix addition first!)





Another Approach to Matrix Multiplication...

$$\begin{matrix} n \\ n \end{matrix}\left[\begin{pmatrix} A & B \\ C & D \end{pmatrix}\right.\quad \overset{n}{\begin{pmatrix} E & G \\ F & H \end{pmatrix}} =$$

$$\begin{matrix} n \\ n \end{matrix}\left[\begin{matrix} \overset{n}{\overset{\downarrow j}{\begin{pmatrix} R = AE+BF & S = AG+BH \\ T = CE+DF & U = CG+DH \end{pmatrix}}} \end{matrix}\right.$$